## Amendments to the Specification

Please replace the paragraph that begins on Page 9, line 14 and carries over to Page 10, line 9 with the following marked-up replacement paragraph:

-- In this situation, the validation of document [[600]] 700 must use the most-specific schema extension, in order to avoid generating exceptions for those attributes that have been added to the base schema. In many cases, the consumer application may not want all of these attribute values, and in fact, receiving the values from the parser may cause problems in the consumer application if it is not adapted for dealing with those attributes (as was noted earlier). Suppose that some consumer application needs (or can process, when present) the gender and age attributes, but does not know about (and therefore cannot use) the marital status attribute. If the objects delivered to this consumer application from the parser were created according to the most-specific schema extension, the parser will not generate syntax errors or exceptions when parsing document [[600]] 700, but the consumer application will receive an attribute value (i.e., marital status) that it does not recognize. This "extra" attribute may cause the application to fail. Or, programmers may have to write additional error checking logic to deal with such unexpected input values. If, on the other hand, the parsing is performed according to the next-most-specific schema extension (i.e., including the gender and age attributes), then the parser will generate a syntax error during the validation process when it encounters a person element with a "maritalStatus" attribute. This may prevent the consumer application from receiving any of the data for the element that has been flagged by the parser as having invalid syntax, which is obviously an undesirable result. --

Please replace the paragraph that begins on Page 20, line 18 and carries over to Page 21, line 12 with the following marked-up replacement paragraph:

-- The setFeature method used by preferred embodiments is preferably implemented by subclassing the existing parse method to provide a feature-based approach. The existing parse method is therefore automatically overridden. This new setFeature invocation takes as parameters two string values (which are illustrated at 1131 and 1132). The first parameter~~ of the~~ is a fully-qualified URI that informs the parser method that abstraction is to be performed, and the second parameter is then a string that identifies the name space of the desired abstraction level (i.e., the name of the schema definition to use when casting objects from the parsed elements). Accordingly, this abstraction level is set as a feature of the parser instance, and the overridden superclass is invoked as usual. See 1140, where the parse method is invoked on this parser instance. (Parameters provided on the invocation have not been shown, but typically identify the input document and where to print any error messages.) The overridden parse method recognizes that the feature has been set, and retrieves the name that is specified for the desired abstraction level and passes that name to the superclass upon invocation. The superclass then uses that abstraction level. --

Please replace the paragraph that begins on Page 23, line 18 and carries over to Page 24, line 11 with the following marked-up replacement paragraph:

-- Commonly-assigned U. S. Patent Application number 10/016,933 (attorney docket RSW920010220US1<u>; now abandoned</u>), which is entitled "Generating Class Library to Represent Messages Described in a Structured Language Schema", discloses techniques whereby class libraries are programmatically generated from a schema. Templates are used for generating code of the class libraries. According to techniques disclosed therein, optional migration logic can be programmatically generated to handle compatibility issues between multiple versions of an XML schema from which class libraries are generated. Multiple versions of an XML schema are read and compared, and a report of their differences is prepared. The differences are preferably used to generate code that handles both the original schema and the changed version(s) of the schema. The class library is then preferably programmatically re-generated such that it includes code for the multiple schema versions. This allows run-time functioning of code prepared according to any of the schema versions. The techniques disclosed therein are not directed toward casting objects at selectable levels. --